Optimization on Microcomputers
The Nelder-Mead Simplex Algorithm[1]

by

J. E. Dennis Jr.[2]
Daniel J. Woods[3]

Technical Report 85-9, December 1985

---

[1]This paper was presented at the ARO Workshop on Microcomputers in Delaware, June, 1985.

[2]Mathematical Sciences Department, Rice University, Houston, Texas 77251. Research sponsored by NSF MCS81-16779, DOE DE-AS05-82ER13016, ARO DAAG-29-83-K-0035, and AFOSR 85-0243.

[3]Mathematical Sciences Department, Rice University, Houston, Texas 77251. Research sponsored by AFOSR 85-0243.

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
DEC 1985		**00-00-1985 to 00-00-1985**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
Optimization on Microcomputers. The Nelder-Mead Simplex Algorithm	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Computational and Applied Mathematics Department ,Rice University,6100 Main Street MS 134,Houston,TX,77005-1892**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**		**11**	

Abstract

In this paper we describe the Nelder-Mead simplex method for obtaining the minimizer of a function. The Nelder-Mead algorithm has several properties that make it a natural choice for implementation and utilization on microcomputers. Stopping criteria for the method are presented as well as a brief discussion of the convergence properties of the method. An algorithmic statement of the method is included as an appendix.

1. Introduction. We consider the problem

$$\underset{x\in\mathbb{R}^n}{\text{minimize}} \quad f(x) \tag{1.1}$$

where $f:\mathbb{R}^n \rightarrow \mathbb{R}^1$ and the problem is to be solved on a microcomputer. The fact that a microcomputer is being used and that problem (1.1) is solvable on this microcomputer leads us to make several assumptions about the problem and the solution environment. First, we assume the amount of storage is small and, therefore, the number of variables, i.e. n, is also small. Additionally, we assume that computing derivatives of the function is not feasible.

There are a class of methods, called direct search methods, see Swann [6] or Brent [1], that attempt to solve problem (1.1) using only function value information. One particular direct search method that is used quite frequently is the Nelder-Mead simplex method presented by Nelder and Mead [3]. Additional references and several modifications of the algorithm are discussed in Parkinson and Hutchinson [5] and Olsson and Nelson [4]. The original method of Nelder and Mead is best-suited for our purposes.

The properties of the Nelder-Mead algorithm that make it appropriate for our problem and environment are its robustness, its simplicity in programming and its low overhead in storage and computation. We say the algorithm is robust because it is very tolerant of noise in the function values. Therefore, the function need not be computed exactly and it may be possible to obtain an approximate function value using many fewer floating point computations.

As we shall see in the following section, the algorithm is very simple to program. Trial points are obtained using very simple algebraic manipulations and these points are accepted or rejected based only on their function values. Also, when the number of variables is small, this algorithm is often competitive with much more complex algorithms that require a great deal of overhead in storage and algebraic manipulations. The low overhead and basic simplicity of this algorithm make it a natural choice for use on microcomputers. An algorithmic specification of the method is given in the Appendix.

2. Algorithm. At each iteration of the Nelder-Mead simplex algorithm, $n+1$ points, denoted by $x_1, x_2, \cdots, x_{n+1}$, are used to compute trial steps. We will often refer to certain of these points based on the order induced by their function values, that is, at the k^{th} iteration we have

$\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_{n+1}$, with $f(\mathbf{x}_1) \leq f(\mathbf{x}_2) \leq \cdots \leq f(\mathbf{x}_{n+1})$. A trial step is accepted or rejected based on the function value of the trial point and the three function values $f(\mathbf{x}_1)$, $f(\mathbf{x}_n)$, and $f(\mathbf{x}_{n+1})$.

The $n+1$ points used at an iteration may be thought of as the vertices of an n-dimensional simplex. In $\mathbb{R}^2$, for example, three points determine a triangle. We denote a simplex S_k, with vertices $x_1, x_2, \cdots, x_{n+1}$, by $S_k = \langle x_1, x_2, \cdots, x_{n+1} \rangle$. It is often the case that a specific vertex of a specific simplex is referenced. Thus, the notation $\mathbf{x}_i^k$ is used to indicate the vertex of simplex S_k that has the i^{th} lowest function value. In Figure 2.1 below, if $f(x_1) = 10.0$, $f(x_2) = 7.0$, and $f(x_3) = 3.0$, then we would have $\mathbf{x}_1 \leftarrow x_3$, $\mathbf{x}_2 \leftarrow x_2$, and $\mathbf{x}_3 \leftarrow x_1$.

Trial steps are generated by the operations of reflection, expansion, contraction, and shrinkage. A reflected vertex is computed by reflecting the worst vertex, $\mathbf{x}_{n+1}$, through the centroid of the remaining vertices. Nelder and Mead compute the reflected vertex as

$$x_r = (1+\alpha)\bar{x} - \alpha \mathbf{x}_{n+1}, \tag{2.1}$$

where $\alpha = 1$, and $\bar{x}$ is the centroid defined by

$$\bar{x} = \frac{1}{n} \sum_{i=1}^{n} \mathbf{x}_i.$$

The reflected vertex is accepted if $f(\mathbf{x}_1) \leq f(x_r) < f(\mathbf{x}_n)$, and the next iteration begins with the simplex defined by $\langle \mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n, x_r \rangle$. Note that x_r has not been ordered with respect to the other vertices.

If the reflected vertex has a lower function value than $\mathbf{x}_1$, i.e., $f(x_r) < f(\mathbf{x}_1)$, then the trial step has produced a good point and the step is expanded. The expansion vertex is computed as

$$x_e = \gamma x_r + (1-\gamma)\bar{x}, \tag{2.2}$$

where $\gamma = 2$. The expansion vertex is accepted if $f(x_e) < f(\mathbf{x}_1)$, otherwise the reflected vertex is accepted. Thus, if $f(x_r) < f(\mathbf{x}_n)$, then either the reflected or expanded vertex is accepted and the next iteration begins.

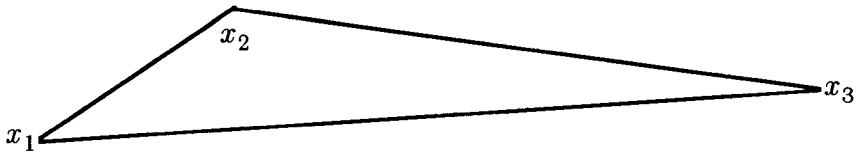

FIGURE 2.1

If the reflected vertex is not a better point than $\mathbf{x}_n$, i.e., $f(\mathbf{x}_n) \leq f(x_r)$, then a contraction step is computed. If the worst vertex is at least as good as the reflected vertex, i.e., $f(\mathbf{x}_{n+1}) \leq f(x_r)$, then the internal contraction vertex is computed as

$$x_c = \beta \mathbf{x}_{n+1} + (1-\beta)\bar{x}, \tag{2.3}$$

otherwise, the external contraction vertex is computed as

$$\hat{x_c} = \beta x_r + (1-\beta)\bar{x}, \tag{2.4}$$

where $\beta = \frac{1}{2}$. The contraction vertex is accepted if it has a lower function value than $\mathbf{x}_n$.

If both the reflection vertex and the contraction vertex are rejected, then the simplex is shrunk. The shrinkage operation is performed by replacing each vertex $\mathbf{x}_i$, except $\mathbf{x}_1$, by the point halfway between $\mathbf{x}_i$ and $\mathbf{x}_1$. This may be written as

$$x_i \leftarrow \frac{(\mathbf{x}_i + \mathbf{x}_1)}{2}. \tag{2.5}$$

Finally, the values $f(x_i)$ are computed and sorted along with $f(\mathbf{x}_1)$. This order determines the simplex $<\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_{n+1}>$ with which the next iteration commences.

If one envisions the simplex sitting on the surface defined by the function, then the operations of the Nelder-Mead algorithm can be thought of as the simplex tumbling down the surface. When the simplex has reached a point where further tumbling is not possible, the simplex contracts, or shrinks towards its lowest point, and the tumbling continues. Figure 2.2 below, illustrates the various trial points for the 2-dimensional simplex $<\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3>$.

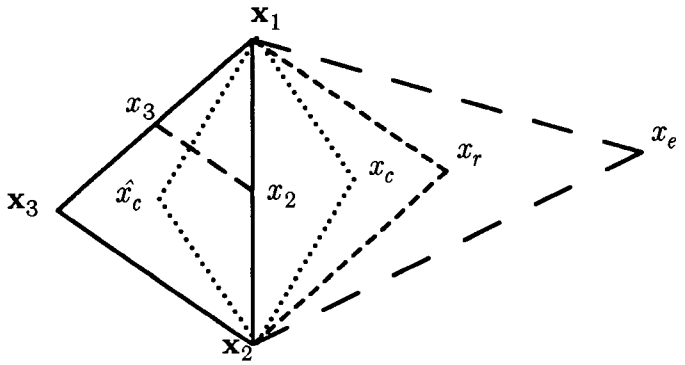

FIGURE 2.2

There are several stopping criteria that have been proposed for this algorithm. Nelder and Mead suggest halting the algorithm when the standard error of the function values falls below some threshold value. That is, the algorithm is halted when the following condition holds:

$$\frac{1}{n}\sum_{i=1}^{n+1}(f(x_i)-\bar{f})^2 < \epsilon_1, \tag{2.6}$$

where $\bar{f}$ is the average of the function values and $\epsilon_1>0$ is some preset value. Parkinson and Hutchinson [5] propose a stopping criterion based on how far the simplex moves at an iteration. They suggest halting the algorithm when the following condition is met:

$$\frac{1}{n}\sum_{i=1}^{n} \|\mathbf{x}_i^k - x_i^{k+1}\|^2 < \epsilon_2, \tag{2.7}$$

where $\|\cdot\|$ is the l_2 norm, $\epsilon_2>0$, and x_i^{k+1} is the i^{th} unordered point in the $k+1^{\text{st}}$ simplex.

Stopping criteria (2.6) and (2.7) are very different. The algorithm is halted in (2.6) based on function value information, while (2.7) uses vertex information. Certain problems can arise with stopping criterion (2.6). For example, if the function values are very close, then the algorithm halts regardless of the size of the simplex. That is, the algorithm may halt when the simplex is very large. For an example of this and additional difficulties with stopping criterion (2.6), see Woods [7].

Objections to using (2.7) as the stopping criterion may also be raised. The main objection to (2.7) is that the left-hand side of (2.7) for a shrinkage step will be greater than the value for a contraction step, and we have observed that shrinkage occurs frequently when the simplex is in a neighborhood of a local minimizer. Woods [7] introduces the stopping criterion

$$\frac{1}{\Delta}\max_{2\le i\le n+1}\|\mathbf{x}_i-\mathbf{x}_1\| \le \epsilon_3, \tag{2.8}$$

where $\Delta = \max(1,\|\mathbf{x}_1\|)$ and $\epsilon_3>0$. This is a measure of the relative size of the simplex. Preliminary testing of (2.8) has indicated that it is a useful stopping criterion for the Nelder-Mead algorithm.

3. Convergence Properties. Although this algorithm is used extensively, the convergence theory is not well-developed. The only convergence results of which we are aware appear in Woods [7], for a slightly modified version of the algorithm, and in the forthcoming paper of Dennis and Woods [2], for the algorithm as stated here. The result of Dennis and Woods states that if the algorithm is applied to a strictly

convex function and the level set of the function corresponding to the value at the worst vertex of the initial simplex is bounded, then the algorithm will converge to a connected set of points, all of which have the same function value. Additionally, each convergent subsequence of the sequence of simplices generated by the algorithm converges to a totally degenerate simplex, i.e., a single point.

Unfortunately, the convergence theory does not provide the desired result of convergence to the minimizer of the strictly convex function. In fact, Dennis and Woods show that this is not necessarily true under their assumptions. They show this by the following example:

EXAMPLE 3.1 : Let $c_1 = (0,32)^T$, $c_2 = (0,-32)^T$, and consider the strictly convex function $f(x) = \frac{1}{2}\max\{\|x-c_1\|^2, \|x-c_2\|^2\}$. The level sets of this function are displayed in Figure 3.1 as is the initial simplex, $S_0 = \langle \mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3 \rangle = \langle (8, 0)^T, (-8, -4)^T, (-16, 10)^T \rangle$. It should be obvious from the figure that both the reflected and contracted vertices are rejected at this iteration and the simplex is shrunk. If the simplex were to shrink at every iteration, then the sequence of simplices would converge to the totally degenerate simplex $S^* = \langle \mathbf{x}_1, \mathbf{x}_1, \mathbf{x}_1 \rangle$, which is not a local minimizer.

Dennis and Woods show that the algorithm can be made to converge to any point $(\alpha,0)^T$ depending upon the choice of the initial simplex. When $\alpha \neq 0$, the algorithm does not converge to the minimizer which is $(0,0)^T$.

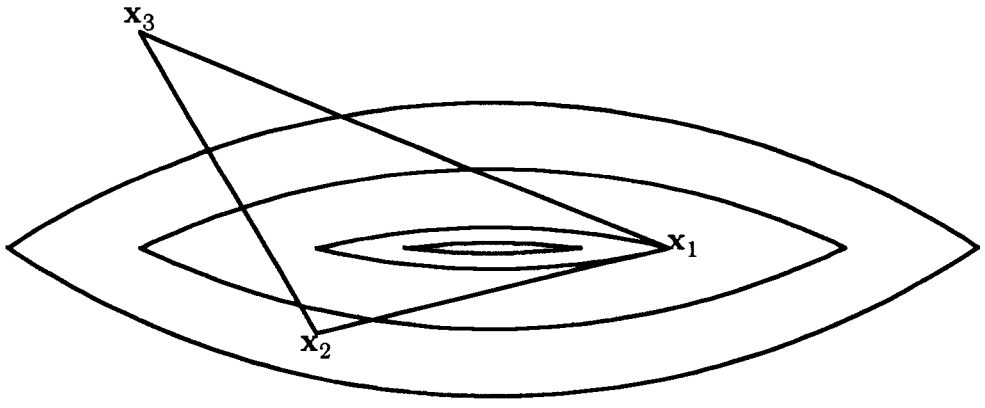

FIGURE 3.1

4. Conclusions. The Nelder-Mead simplex algorithm is very well-suited for use on microcomputers. It is robust, easy to program and requires very little storage and information for execution. Although convergence properties for the algorithm are not well understood, the algorithm is used in many applications.

REFERENCES

[1] R. P. BRENT, *Algorithms for Minimization Without Derivatives*, Prentice-Hall, Englewood Cliffs, N.J., 1973.

[2] J. E. DENNIS JR., D. J. WOODS, *Convergence Properties of the Nelder-Mead Simplex Algorithm*, in preparation.

[3] J. A. NELDER, R. MEAD, *A simplex method for function minimization*, The Computer Journal (1965), Vol. 7, p 308.

[4] D. M. OLSSON, L. S. NELSON, *Nelder-Mead simplex procedure for function minimization*, Technometrics (1975), Vol. 17, p. 45.

[5] J. M. PARKINSON, D. HUTCHINSON, *An investigation into the efficiency of variants of the simplex method*, Numerical Methods for Non-linear Optimization, (F.A. Lootsma, ed.), p. 115, Academic Press, London and New York, 1972.

[6] W. H. SWANN, *Direct search methods*, Numerical Methods for Unconstrained Optimization, (W. Murray, ed.), p. 13, Academic Press, London and New York, 1972.

[7] D. J. WOODS, *An Interactive Approach for Solving Multi-Objective Optimization Problems*, Available as Technical Report 85-5, Mathematical Sciences Department, Rice University, Houston, TX. 77251, 1985.

Appendix. We now present an algorithmic statement of the Nelder-Mead simplex method. For simplicity, we have introduced the temporary variables x^k and x^t. In an implementation of the algorithm pointers to the corresponding vertices would be used. Various stopping criteria for the algorithm are discussed in Section 2.

Algorithm A-1: Nelder-Mead Simplex Algorithm

Given S_0 with vertices $<\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_{n+1}>$, set $\alpha=1$, $\beta=\frac{1}{2}$, $\gamma=2$.

For k = 1, 2, $\cdots$

set $x_i = \mathbf{x}_i, \quad i=1, \cdots, n$

compute $\bar{x} = \frac{1}{n} \sum_{i=1}^{n} \mathbf{x}_i$

compute $x_r = (1+\alpha)\bar{x} - \alpha \mathbf{x}_{n+1}$

$x^k = x_r$

if $(f(x_r) < f(\mathbf{x}_n))$ **then**

if $(f(x_r) < f(\mathbf{x}_1))$ **then**

compute $x_e = \gamma x_r + (1-\gamma)\bar{x}$

if $(f(x_e) < f(\mathbf{x}_1))$ **then** $x^k = x_e$

else set $x^t = \mathbf{x}_{n+1}$

if $(f(x_r) < f(x^t))$ **then** $x^t = x_r$

compute $x_c = \beta x^t + (1-\beta)\bar{x}$

if $(f(x_c) < f(\mathbf{x}_n))$ **then** $x^k = x_c$

else $x_j = \frac{\mathbf{x}_1+\mathbf{x}_j}{2}$ for $j = 2, \cdots, n$

$x^k = \frac{\mathbf{x}_1+\mathbf{x}_{n+1}}{2}$

Check the stopping criterion.

Sort $f(x_1), f(x_2), \cdots, f(x_n), f(x^k)$ to

obtain $S_k = <\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_{n+1}>$.